RUNNING CODE EXAMPLE: Liskov Substitution Principle (LSP)

Nama : Flavianus Putratama

NIM : 21120122140105

Kelas : RPLBK – Kelas C

Focus Group : C

A. Problem : Persegi Panjang vs. Persegi

Kita memiliki sebuah superclass `Rectangle` (persegi panjang) dan subclass `Square` (persegi). Kita akan melihat bagaimana penggunaan subclass `Square` untuk menggantikan `Rectangle` dapat melanggar LSP.

```
# Superclass Rectangle
class Rectangle:
    def __init__(self, width, height):
        self._width = width
        self._height = height

    def set_width(self, width):
        self._width = width

    def set_height(self, height):
        self._height = height

    def get_area(self):
        return self._width * self._height

# Subclass Square yang melanggar LSP
class Square(Rectangle):
    def __init__(self, side):
        super().__init__(side, side)

    # Override set_width agar sisi selalu sama
    def set_width(self, width):
        self._width = self._height = width

    # Override set_height agar sisi selalu sama
    def set_height(self, height):
        self._width = self._height = height

# Fungsi untuk menghitung area dari objek Rectangle
def calculate_area(rectangle: Rectangle):
    rectangle.set_width(5)
    rectangle.set_height(10)
    print(f"Area: {rectangle.get_area()}")

# Testing
rect = Rectangle(2, 3)
calculate_area(rect)  # Output: Area: 50 (sesuai dengan
panjang dan lebar)

square = Square(5)
```

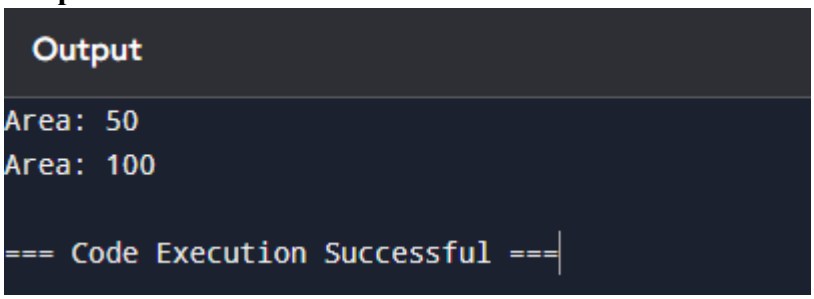
```
calculate_area(square)  # Output: Area: 100 (tidak sesuai dengan
yang diharapkan)
```

Penjelasan:

a. Kelas `Rectangle` memiliki dua metode untuk mengatur lebar dan tinggi, yang bekerja secara independen.

b. Kelas `Square` melanggar LSP karena ketika kita mengatur lebar (`set_width`) atau tinggi (`set_height`), kedua sisi harus selalu sama. Ini menyebabkan perilaku yang tidak diharapkan saat menghitung area di fungsi `calculate_area()`. Misalnya, setelah memanggil `set_width(5)` dan `set_height(10)`, kita mengharapkan area menjadi `50`, tetapi karena `Square` memperlakukan lebar dan tinggi sama, area sebenarnya adalah `100`.

B. Output



C. Solver

Untuk memperbaiki pelanggaran LSP, kita harus membuat kelas `Square` dan `Rectangle` terpisah, tanpa membuat `Square` menjadi subclass dari `Rectangle`, karena mereka memiliki perilaku yang berbeda.

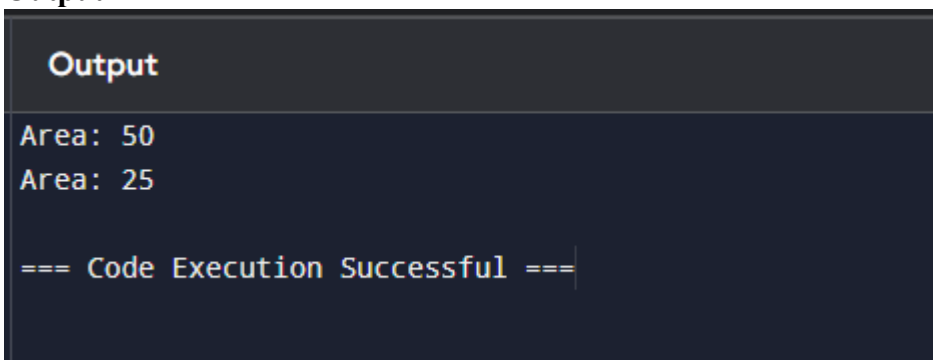
```
# Superclass Shape (bentuk umum)
class Shape:
    def get_area(self):
        raise NotImplementedError("This method should be
overridden")

# Subclass Rectangle
class Rectangle(Shape):
    def __init__(self, width, height):
        self._width = width
        self._height = height
```

```
            def set_width(self, width):
                self._width = width

            def set_height(self, height):
                self._height = height

            def get_area(self):
                return self._width * self._height

        # Subclass Square
        class Square(Shape):
            def __init__(self, side):
                self._side = side

            def set_side(self, side):
                self._side = side

            def get_area(self):
                return self._side * self._side

        # Fungsi untuk menghitung area dari objek Shape
        def calculate_area(shape: Shape):
            print(f"Area: {shape.get_area()}")

        # Testing
        rect = Rectangle(5, 10)
        calculate_area(rect)  # Output: Area: 50

        square = Square(5)
calculate_area(square)  # Output: Area: 25
```

Penjelasan :

a. Dengan kode yang sudah diperbaharui, `Rectangle` dan `Square` adalah subclass dari `Shape`, tanpa hubungan inheritance di antara mereka. Mereka masing-masing mengimplementasikan perilaku area yang sesuai.
b. Fungsi `calculate_area()` sekarang n dapat bekerja untuk `Rectangle` dan `Square` secara terpisah, sesuai dengan sifat masing-masing bentuk.

D. Output



E. A
F. Aa
G. A
H.